

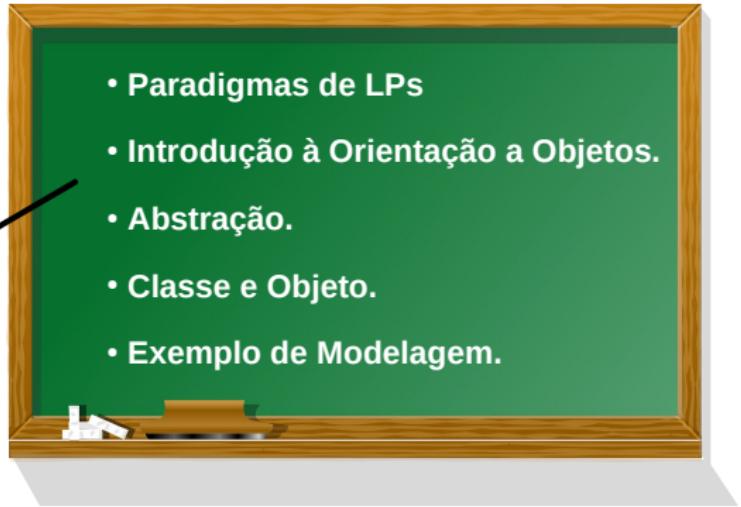
Conceitos Básicos de Orientação a Objetos

Luiz Henrique de Campos Merschmann
Departamento de Computação Aplicada
Universidade Federal de Lavras

luiz.hcm@ufla.br



Na Aula de Hoje



Paradigmas de LPs

- ▶ As LPs podem ser classificadas quanto ao paradigma de programação.
- ▶ O que é um paradigma?
 - ▶ Um conjunto de teorias, padrões e métodos que juntos representam uma forma de organizar o conhecimento.
 - ▶ Um paradigma é um padrão de resolução de problemas.
- ▶ Algumas LPs são projetadas para suportar mais de um paradigma.
- ▶ Quais paradigmas de LP você conhece?

Paradigmas de LPs

Paradigma Imperativo

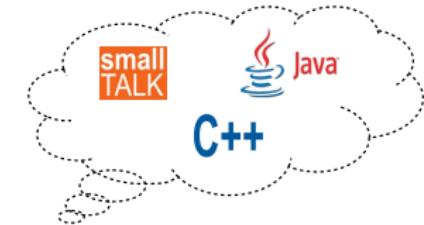


Estruturado

Paradigma Declarativo



Lógico



Orientado a Objetos



Funcional

Paradigma Imperativo

Paradigma Estruturado

- ▶ Pensamento de programação mais voltado ao pensamento de máquina.
- ▶ As linguagens estruturadas são relativamente fáceis de serem aprendidas.
 - ▶ Muito usadas para se aprender a programar.
- ▶ Para problemas simples e diretos ainda é a melhor solução.

Paradigma Imperativo

Paradigma Estruturado

- ▶ Tem como foco principal as ações.
- ▶ Os programas em linguagem estruturada usam:
 - ▶ Variáveis.
 - ▶ Procedimentos e funções como mecanismos de estruturação.
- ▶ Exemplo:

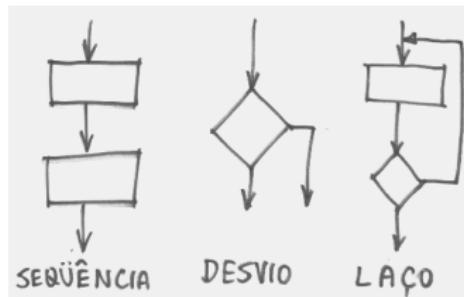
```
1 //Exemplo de variáveis
2 int a,b;
3 string nome;
4 char sexo;
5
6 //Exemplo de função
7 int Multiplicacao(int a, int b){
8     int mult;
9     mult = a * b;
10    return mult;
11 }
```

Paradigma Imperativo

Paradigma Estruturado

- ▶ Os programas podem ser reduzidos a apenas três estruturas:

- ▶ Sequência.
- ▶ Decisão (desvio).
- ▶ Repetição (laço).

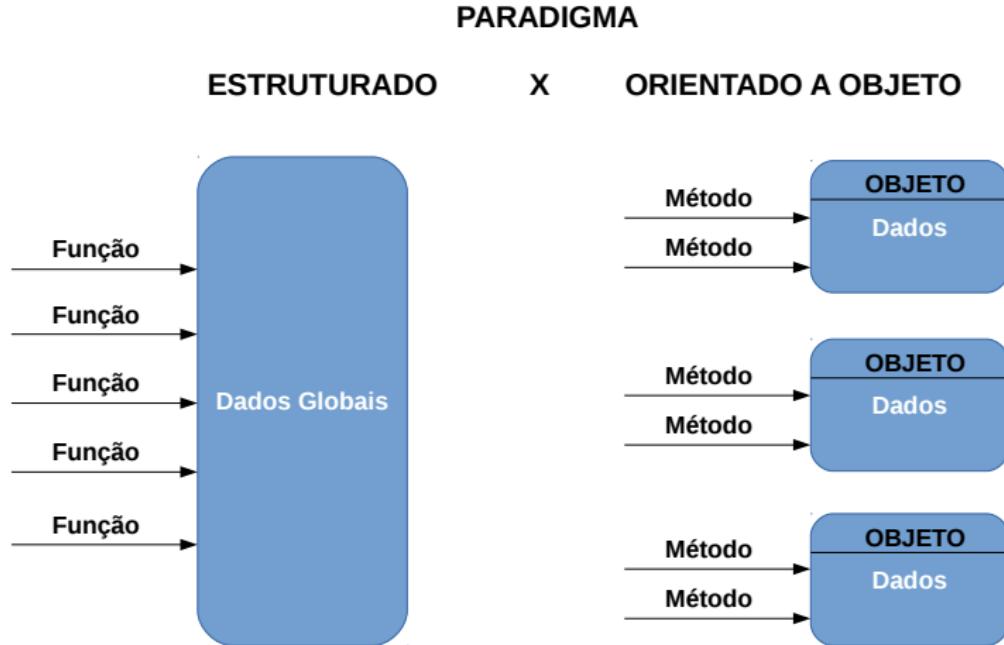


Paradigmas de LPs

Paradigma Orientado a Objetos

- ▶ Alguns autores não consideram um paradigma propriamente dito, mas sim uma extensão do paradigma estruturado.
- ▶ Evolução da programação estruturada.
- ▶ Pensamento de programação mais voltado ao pensamento humano.
- ▶ Conceitos envolvidos: classes, objetos, atributos, métodos, herança etc.
- ▶ Um programa é uma coleção de objetos que interagem entre si, passando mensagens que transformam o seu estado.

Paradigmas de LPs



Orientação a Objetos

Orientação a Objetos

Introdução

- ▶ O conceito de Orientação a Objetos data do final da década de 60 e início da década de 70.
 - ▶ Simula 67 (60's).
 - ▶ Smalltalk (70's).
 - ▶ C++ (80's).
- ▶ Surgiu da necessidade de modelar sistemas mais complexos.

Orientação a Objetos

Como modelar o mundo real usando um conjunto de componentes de software?



- ▶ Como vocês descreveriam essa cena?
- ▶ Ou seja, nós compreendemos o mundo através de objetos!

Orientação a Objetos



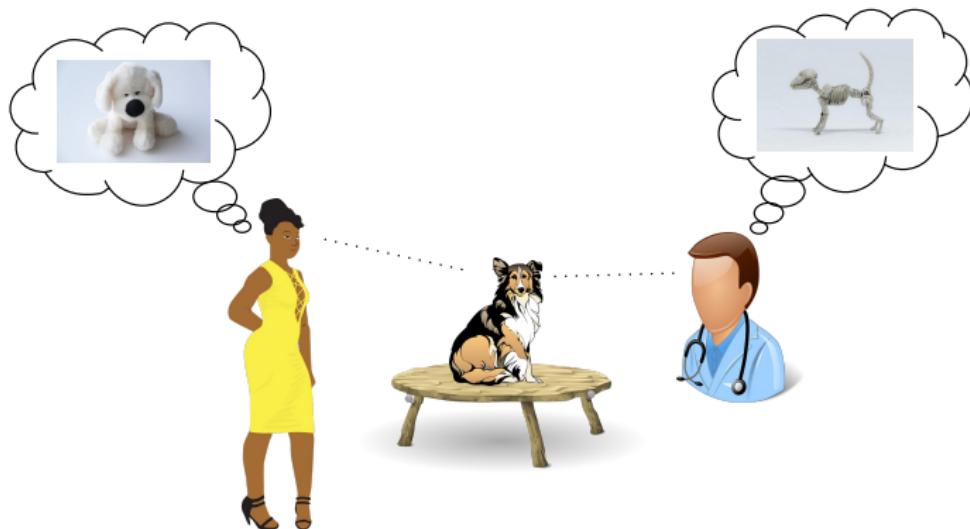
Portanto, em POO nós **inicialmente**
pensamos nos objetos que
fazem parte do problema.

Nós não começamos pensando
nas estruturas de dados
ou rotinas.

Orientação a Objetos

Abstração

- ▶ É um dos pilares da POO.
- ▶ Consiste em trabalhar um objeto se preocupando somente com suas principais propriedades, sem se apegar a detalhes desnecessários.



A abstração diz respeito ao foco nas características essenciais de um objeto de acordo com a perspectiva do observador.

Orientação a Objetos

Introdução

- ▶ A ideia é modelar utilizando objetos, determinando como eles devem se comportar e como devem interagir entre si.
- ▶ Esse paradigma de programação tenta ser o mais óbvio, natural e exato possível.
- ▶ Conceitos essenciais:
 - ▶ Classes e Objetos.
 - ▶ Atributos, Métodos e Mensagens.
 - ▶ Herança e Associação.
 - ▶ Encapsulamento.
 - ▶ Polimorfismo.
 - ▶ Interfaces.

Conceitos de Orientação a Objetos

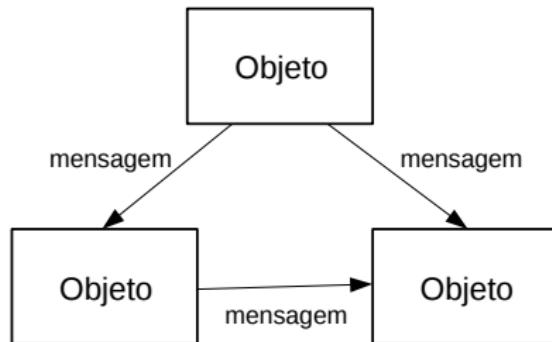
Objetos

- ▶ São elementos computacionais que representam alguma entidade (abstrata ou concreta) do domínio de interesse do problema sob análise.
- ▶ Correspondem a uma abstração para algo do mundo real.
- ▶ Portanto, qualquer elemento conceitual de um problema pode ser representado como um objeto em um programa (p.ex.: carros, pessoas, empréstimos etc.)

Conceitos de Orientação a Objetos

Objetos

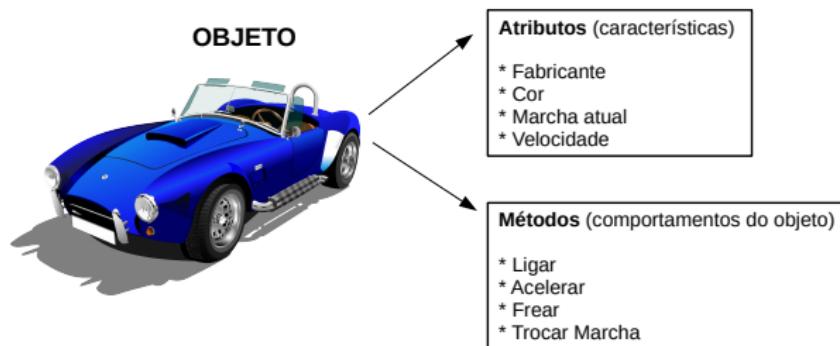
- ▶ Um programa é uma coleção de objetos que interagem entre si.
- ▶ Para fazer uma requisição a um objeto envia-se uma **mensagem** para o mesmo.
 - ▶ Mensagem = chamada a um método pertencente a um objeto.



Conceitos de Orientação a Objetos

Objetos

- ▶ Os objetos possuem:
 - ▶ Atributos: são as características que o definem.
 - ▶ Análogos às variáveis em programação estruturada.
 - ▶ Métodos: determinam o comportamento dos objetos.
 - ▶ Análogos às funções ou procedimentos da programação estruturada



Conceitos de Orientação a Objetos

Objetos

- Estado: conjunto dos valores dos atributos em um determinado momento.

- **Objeto:** Ventilador
 - Nº de pás: 4
 - Tipo de pás: madeira
 - Cor: mogno
 - Nº de velocidades: 3
 - Função exaustor: sim



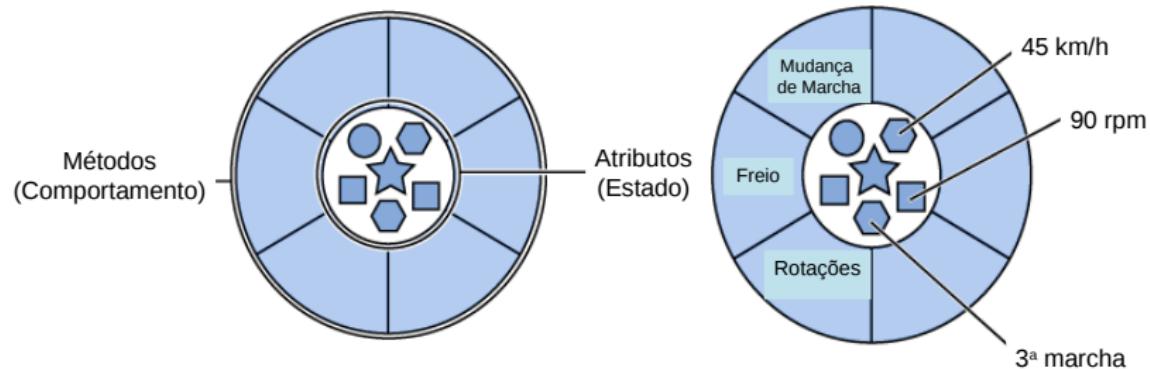
- O comportamento do objeto é definido pela forma como ele age e reage, em termos de mudança de seu estado e o relacionamento com os demais objetos do sistema.

- **Objeto:** Ventilador
 - Alterar o nº de pás.
 - Solicitar o nº de pás.
 - Alterar a cor.
 - Solicitar a cor.
 - Ligar modo exaustor.
 - Aumentar velocidade.



Conceitos de Orientação a Objetos

Objetos



Conceitos de Orientação a Objetos

Classe



- ▶ No mundo real, encontramos vários objetos de um mesmo tipo.

Conceitos de Orientação a Objetos

Classe



- ▶ Uma classe é como um “projeto” a partir do qual objetos individuais são criados.

Conceitos de Orientação a Objetos

Classe

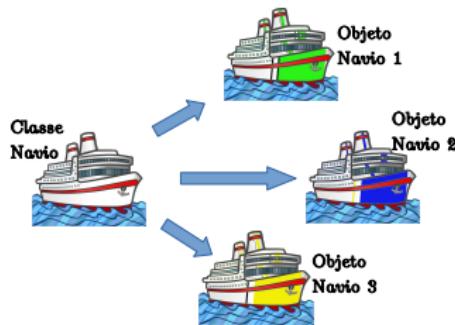
- ▶ Classe: é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações e relacionamentos.
- ▶ Para definir uma classe é necessário **abstrair um conjunto de objetos** com características similares.
- ▶ A definição de uma classe corresponde à formalização de um **tipo de dado** e todas as operações associadas a esse tipo.



Conceitos de Orientação a Objetos

Classe *versus* Objeto

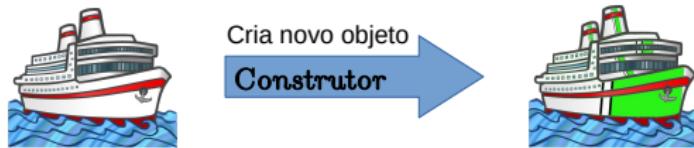
- ▶ Um objeto é uma **instância** da classe de objetos.
- ▶ Um objeto constitui uma entidade concreta com tempo e espaço de existência, enquanto a classe é somente uma abstração, ou seja:
 - ▶ As **classes** “existem” no **código-fonte** do programa.
 - ▶ Já os **objetos** existem apenas durante a **execução do programa**.
- ▶ Instanciação de objeto = criação



Conceitos de Orientação a Objetos

Classe

- ▶ Outros possíveis membros de uma classe são:
 - ▶ Construtores: definem as operações a serem realizadas quando um objeto é criado.



- ▶ Destruidores: definem as operações a serem realizadas quando um objeto é destruído.



Colocando a Mão na Massa . . .



- ▶ Defina as classes, com seus respectivos atributos e métodos para o seguinte cenário:
- ▶ Pizzaria.

Exemplo da Pizzaria

- ▶ Foi dada muito pouca informação sobre o funcionamento da Pizzaria, portanto, as ideias das classes, atributos e métodos podem variar muito.
- ▶ Em uma situação real seria feito um levantamento detalhado “das regras do negócio”. E isso deixaria as coisas mais claras.
- ▶ De qualquer forma, a situação é interessante para exercitarmos algumas questões.

Dica 1



Uma dúvida muito comum é não ter certeza se algo é um objeto ou só a característica (atributo) de um objeto.

- ▶ Levante todos os dados que lembrar e depois veja se algum dado pode ser encaixado como atributo de outro.

Exemplo da Pizzaria

Exemplo de dados levantados:

- ▶ Pizzas, sabores, preços da pizza, cliente, nome, telefone e endereço de clientes, nome de vendedores, bebidas, com borda, quantidade de pizzas, quantidade de bebidas, pedidos, total do pedido, bebida, preço da bebida, taxas de entrega.

Seguindo a **Dica 1**, vamos procurar o que é característica de outra coisa.

- ▶ Características de Pizza: sabor, preço de uma pizza, com borda.
- ▶ Características da Bebida: tipo, preço da bebida.
- ▶ Características do Cliente: nome, telefone e endereço.
- ▶ Características do Vendedor: nome.
- ▶ Características do Pedido: pizzas, quantidade de cada pizza, bebidas, quantidade de cada bebida, total do pedido, cliente, vendedor e taxa de entrega.

Exemplo da Pizzaria



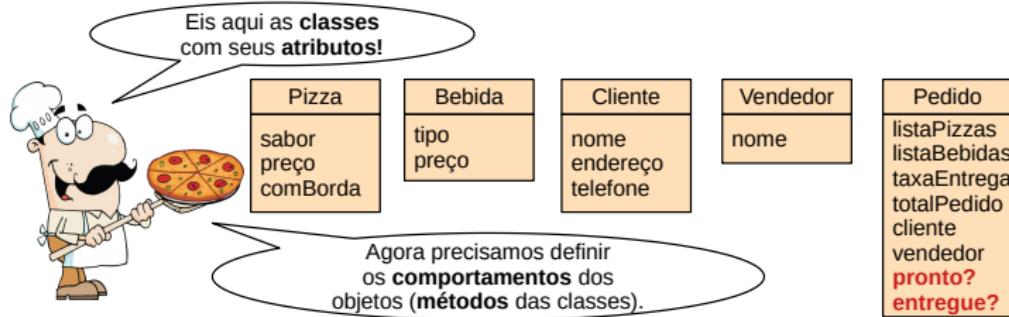
Dica 2

Tudo que foi identificado como tendo características seriam objetos, e as características são seus atributos. Portanto, definiríamos as classes Pizza, Bebida, Cliente, Vendedor e Pedido.

Observação

Veja que a classe Pedido teria como atributos objetos de outras classes (Pizza, Bebida, Cliente e Vendedor). E realmente uma classe pode ter tanto atributos básicos (*int, float...*) como também atributos que são objetos de outras classes.

Exemplo da Pizzaria



Podemos ter os seguintes comportamentos:

- ▶ Nas classes Pizza e Bebida: alterar preço.
- ▶ Na classe Cliente: alterar telefone, alterar endereço.
- ▶ Na classe Vendedor: podemos considerar que os objetos não são alterados depois de criados.
- ▶ Na classe Pedido: adicionar pizza, adicionar bebida, definir cliente, definir vendedor, calcular taxa de entrega, calcular total do pedido, **indicar pedido pronto, indicar pedido entregue. Opa! Precisamos** adicionar os atributos para armazenar o *status* do pedido.

Exemplo da Pizzaria

Mas quem cria um objeto da classe Pedido?

- ▶ Precisamos de, pelo menos, uma classe que “controle” o funcionamento do programa.

O nosso sistema é para controlar o que?

- ▶ Uma pizzaria! Desse modo, vamos criar uma classe para representar a Pizzaria e ser a nossa classe de controle.
- ▶ Quais seriam seus atributos e métodos?



Dica 3

Na classe de “controle” pensamos em todas as ações que o usuário do sistema pode realizar.

Pizzaria
nome
listaClientes
listaVendedores
listaBebidas
listaPizzas
listaPedidos
cadastrarCliente
cadastrarVendedor
cadastrarBebida
cadastrarPizza
fazerPedido
alterarDadosCliente
alterarDadosPizza
alterarDadosBebida
definirPedidoPronto
definirPedidoEntregue
cancelarPedido
:

Exemplo da Pizzaria

Observação

- ▶ Note que os métodos de cadastro ficam em uma classe de controle e não na classe associada ao objeto que se deseja cadastrar.
- ▶ Exemplo: o cadastro de clientes não pode ficar na classe Cliente.

A maneira mais fácil de entender isso é lembrar que um objeto da classe Cliente, representa um cliente específico, por ex., o João. Se o cadastro de clientes fosse na classe Cliente, para cadastrar um outro cliente (ex: Maria), teríamos que fazer *joao.cadastrarCliente("Maria", ...)*. Mas não faz sentido um cliente ter a ação de cadastrar outro!

Exemplo da Pizzaria



Dica 4

Geralmente separamos a interação com o usuário (tela ou menu) da “regra de negócio”. Dessa forma, teríamos que fazer mais uma classe para representar essa interação com o usuário (separada da classe Pizzaria).

Exemplo da Pizzaria



Pizza	Bebida
sabor preço comBorda	tipo preço
alterarPreço	alterarPreço
Cliente	Vendedor
nome endereço telefone	nome
alterarEndereço alterarTelefone	

Pedido	Pizzaria
listaPizzas listaBebidas taxaEntrega totalPedido cliente vendedor pronto? entregue? adicionarPizza adicionarBebida definirCliente definirVendedor calcularTaxaEntrega calcularTotalPedido indicarPedidoPronto indicarPedidoEntregue	nome listaClientes listaVendedores listaBebidas listaPizzas listaPedidos cadastrarCliente cadastrarVendedor cadastrarBebida cadastrarPizza fazerPedido alterarDadosCliente alterarDadosPizza alterarDadosBebida definirPedidoPronto definirPedidoEntregue cancelarPedido
	:

Atividade



- ▶ Defina as classes, com seus respectivos atributos e métodos para os seguinte cenário:
 - ▶ Imobiliária.

Perguntas?

